```
RRRRRRRRRRR      MMM          MMM      SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRRRRRRRRRR      MMM          MMM      SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSS
RRR    RRR       MMM          MMM               SSS
RRR    RRR       MMM          MMM               SSS
RRR    RRR       MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
```

FILEID**NTOCLOSE

```
NN      NN  TTTTTTTTTT    000000      CCCCCCCC  LL            000000      SSSSSSSS  EEEEEEEEEE
NN      NN  TTTTTTTTTT    000000      CCCCCCCC  LL            000000      SSSSSSSS  EEEEEEEEEE
NN      NN      TT      00      00  CC        LL          00      00  SS            EE
NN      NN      TT      00      00  CC        LL          00      00  SS            EE
NNNN    NN      TT      00    0000  CC        LL          00      00  SS            EE
NNNN    NN      TT      00    0000  CC        LL          00      00  SS            EE
NN  NN  NN      TT      00  00  00  CC        LL          00      00    SSSSSS      EEEEEEEE
NN  NN  NN      TT      00  00  00  CC        LL          00      00    SSSSSS      EEEEEEEE
NN    NNNN      TT      0000    00  CC        LL          00      00          SS  EE
NN    NNNN      TT      0000    00  CC        LL          00      00          SS  EE
NN      NN      TT      00      00  CC        LL          00      00          SS  EE
NN      NN      TT      00      00  CC        LL          00      00          SS  EE
NN      NN      TT        000000      CCCCCCCC  LLLLLLLLLL    000000    SSSSSSSS    EEEEEEEEEE
NN      NN      TT        000000      CCCCCCCC  LLLLLLLLLL    000000    SSSSSSSS    EEEEEEEEEE

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          $BEGIN  NTOCLOSE,000,NF$NETWORK,<NETWORK CLOSE FILE>
0000     2
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27
0000    28 ;++
0000    29 ;
0000    30 ; Facility: RMS
0000    31 ;
0000    32 ; Abstract:
0000    33 ;
0000    34 ;       This module communicates with the File Access Listener (FAL) at the
0000    35 ;       remote node to close the specified file.
0000    36 ;
0000    37 ; Environment: VAX/VMS, executive mode
0000    38 ;
0000    39 ; Author: James A. Krycka,      Creation Date:  09-DEC-1977
0000    40 ;
0000    41 ; Modified By:
0000    42 ;
0000    43 ;       V03-002 JAK0125         J A Krycka      07-SEP-1983
0000    44 ;               When it is necessary to build an Access message to convey
0000    45 ;               certain FOP options on close, construct the file specification
0000    46 ;               from the FWA instead of the NAM block, as RMS now keeps the
0000    47 ;               FWA around for the duration of the file access.
0000    48 ;
0000    49 ;--
```

```
                        .SBTTL  DECLARATIONS

;
; Include Files:
;

        $DAPPLGDEF                      ; Define DAP prologue symbols
        $DAPHDRDEF                      ; Define DAP message header
        $DAPCNFDEF                      ; Define DAP Configuration message
        $DAPATTDEF                      ; Define DAP Attributes message
        $DAPACCDEF                      ; Define DAP Access message
        $DAPCMPDEF                      ; Define DAP Access Complete message
        $IFBDEF                         ; Define IFAB symbols
        $NWADEF                         ; Define Network Work Area symbols

;
; Macros:
;
;       None
;
; Equated Symbols:
;

        ASSUME  DAP$Q_DCODE_FLG EQ 0
        ASSUME  NWA$Q_FLG EQ 0

;
; Own Storage:
;
;       None
;
```

```
0000     83          .SBTTL  NT$CLOSE - PERFORM NETWORK CLOSE FUNCTION
0000     84
0000     85 ;++
0000     86 ; NT$CLOSE - engages in a DAP dialogue with the remote FAL to close the
0000     87 ;       specified file and to process selected FOP options and XABs.
0000     88 ;
0000     89 ; Calling Sequence:
0000     90 ;
0000     91 ;       BSBW    NT$CLOSE
0000     92 ;
0000     93 ; Input Parameters:
0000     94 ;
0000     95 ;       R8      FAB address
0000     96 ;       R9      IFAB address
0000     97 ;       R10     IFAB address
0000     98 ;       R11     Impure Area address
0000     99 ;
0000    100 ; Implicit Inputs:
0000    101 ;
0000    102 ;       User PRO and RDT XABs
0000    103 ;       DAP$L_CRC_RSLT
0000    104 ;       DAP$V_DAPCRC
0000    105 ;       DAP$V_FOPDLT, FOPSCF, FOPSPL
0000    106 ;       DAP$V_VAXVMS
0000    107 ;       IFB$V_DLT, DMO, RWC, SCF, SPL
0000    108 ;       IFB$L_FWA_PTR
0000    109 ;       IFB$L_NWA_PTR
0000    110 ;       NWA$V_FTM_EOF
0000    111 ;       NWA$V_FTM_RETRV
0000    112 ;
0000    113 ; Output Parameters:
0000    114 ;
0000    115 ;       R0      Status code (RMS)
0000    116 ;       R1-R7   Destroyed
0000    117 ;       AP      Destroyed
0000    118 ;
0000    119 ; Implicit Outputs:
0000    120 ;
0000    121 ;       None
0000    122 ;
0000    123 ; Completion Codes:
0000    124 ;
0000    125 ;       Standard RMS completion codes
0000    126 ;
0000    127 ; Side Effects:
0000    128 ;
0000    129 ;       Optionally, a file may be deleted, executed, printed, submitted
0000    130 ;       (executed then deleted), or spooled (printed then deleted) on close
0000    131 ;       at the remote node. Note that there is a distinction between the DAP
0000    132 ;       execute and submit functions.  Furthermore, the DCL SUBMIT/REMOTE
0000    133 ;       command really maps into the DAP execute function (i.e., the file is
0000    134 ;       not deleted after the batch job completes). Also, the file's protection
0000    135 ;       and revision information may be optionally altered on close.
0000    136 ;
0000    137 ;       If an RMS$_CRC error is reported by FAL, then a message will be sent
0000    138 ;       to the DECnet Event Logger (EVL).
0000    139 ;--
```

```
                     0000   141 NT$CLOSE::                                  ; Entry point
                     0000   142         $TSTPT  NTCLOSE                     ;
 57   3C A9   D0      0006   143         MOVL    IFB$L_NWA_PTR(R9),R7        ; Get address of NWA (and DAP)
                     000A   144
                     000A   145 ;+
                     000A   146 ; If the file was opened for write access then XABs are input on close. If so
                     000A   147 ; build a mask (NWA$W_BUILD) that will determine which optional DAP messages
                     000A   148 ; to send to FAL to modify the attributes of the file on close.
                     000A   149 ; For $CLOSE, these are the TIM and PRO messages.
                     000A   150 ;-
                     000A   151
                     000A   152 BUILD_MASK:                                 ; Build NWA$W_DISPLAY
 69   30   E1          000A   153         BBC     #IFB$V_WRTACC,(R9),-        ; Ignore XABs if file was opened for
      53               000D   154                 SEND_CMP_CLOSE              ;  for read-only access
 56   01   D0          000E   155         MOVL    #1,R6                       ; Signal this as a close operation
    FFEC'  30          0011   156         BSBW    NT$SCAN_XABCHN              ; Scan user XAB chain and check FAL's
                     0014   157                                             ;  capabilities; request mask put in R2
   44 50   E9          0014   158         BLBC    R0,EXIT                     ; Branch on failure to complete scan
 52   0F   AA          0017   159         BICW2   #<<DAP$M_DSP_ATT>!-         ; Remove these messages from build mask
                     001A   160                   <DAP$M_DSP_ALL>!-         ;  as their corresponding XABs are
                     001A   161                   <DAP$M_DSP_KEY>!-         ;  not input to the close operation
                     001A   162                   <DAP$M_DSP_SUM>!-         ;
                     001A   163                   0>,R2                     ;
00D2 C7   52   B0      001A   164         MOVW    R2,NWA$W_BUILD(R7)          ; Save build mask
      40   13          001F   165         BEQL    SEND_CMP_CLOSE              ; Branch if no pertinent XABs on chain
                     0021   166
                     0021   167 ;+
                     0021   168 ; Build and send DAP Access Complete (change-begin) message to partner.
                     0021   169 ;-
                     0021   170
                     0021   171 SEND_CMP_BEGIN:                             ; (required message to modify file)
 50   07   D0          0021   172         MOVL    #DAP$K_CMP_MSG,R0           ; Get message type value
    FFD9'  30          0024   173         BSBW    NT$BUILD_HEAD               ; Construct message header
 85   06   90          0027   174         MOVB    #DAP$K_CHANGE_B,(R5)+       ; Store CTLFUNC field
    FFD3'  30          002A   175         BSBW    NT$BUILD_TAIL               ; Finish building message
    FFD0'  30          002D   176         BSBW    NT$TRANSMIT                 ; Send Access Complete message to FAL
   28 50   E9          0030   177         BLBC    R0,EXIT                     ; Branch on failure
                     0033   178
                     0033   179 ;+
                     0033   180 ; Build and send DAP Date and Time message to partner from the Revision
                     0033   181 ; Date and Time XAB.
                     0033   182 ;-
                     0033   183
                     0033   184 SEND_TIM:                                   ; (optional message)
      04   E1          0033   185         BBC     #DAP$V_DSP_TIM,-            ; Branch if Date and Time message
00D2 C7               0035   186                 NWA$W_BUILD(R7),-           ;  should not be sent
      0E               0038   187                 SEND_PRO                    ;
56 0114 C7   D0          0039   188         MOVL    NWA$L_RDTXABADR(R7),R6      ; Get address of user RDTXAB
    FFBF'  30          003E   189         BSBW    NT$ENCODE_TIM_R             ; Build message
    FFBC'  30          0041   190         BSBW    NT$TRANSMIT                 ; Send Date and Time message to FAL
   14 50   E9          0044   191         BLBC    R0,EXIT                     ; Branch on failure
                     0047   192
                     0047   193 ;+
                     0047   194 ; Build and send DAP Protection message to partner.
                     0047   195 ;-
                     0047   196
                     0047   197 SEND_PRO:                                   ; (optional message)
```

```
                05   E1  0047   198          BBC     #DAP$V_DSP_PRO,-        ; Branch if Protection message
            00D2 C7      0049   199                  NWA$W_BUILD(R7),-        ;  should not be sent
                0F       004C   200                  SEND_CMP_END            ;
     56     0110 C7  D0  004D   201          MOVL    NWA$L_PROXABADR(R7),R6  ; Get address of user PROXAB
               FFAB' 30  0052   202          BSBW    NT$ENCODE_PRO           ; Build message
               FFA8' 30  0055   203          BSBW    NT$TRANSMIT             ; Send Protection message to FAL
              01 50  E8  0058   204          BLBS    R0,SEND_CMP_END         ; Branch on success
                     05  005B   205 EXIT:   RSB                             ; Exit with RMS code in R0
```

```
                     005C    207 ;+
                     005C    208 ; Build and send DAP Access Complete (change-end) message to partner if this is
                     005C    209 ; a change of file attributes on close sequence.
                     005C    210 ;
                     005C    211 ; OR
                     005C    212 ;
                     005C    213 ; Build and send DAP Access Complete (close) message to partner if this is a
                     005C    214 ; normal close operation.
                     005C    215 ;
                     005C    216 ; In either case, FOP options (DLT, SCF, SPL, ...) will be specified in this
                     005C    217 ; message.
                     005C    218 ;-
                     005C    219
                     005C    220 SEND_CMP_END:                               ; (required message to modify file)
51    07    D0       005C    221         MOVL    #DAP$K_CHANGE_E,R1          ; Get function code
      03    11       005F    222         BRB     SEND_CMP                    ; Join common code
                     0061    223 SEND_CMP_CLOSE:                             ; (required message for normal close)
51    01    D0       0061    224         MOVL    #DAP$K_CLOSE,R1             ; Get function code
                     0064    225 SEND_CMP:                                   ; Common code
                     0064    226         $SETBIT #NWA$V_LAST_MSG,(R7)        ; Declare this last message to block
50    07    D0       0068    227         MOVL    #DAP$K_CMP_MSG,R0           ; Get message type value
   FF92'    30       006B    228         BSBW    NT$BUILD_HEAD               ; Construct message header
85    51    90       006E    229         MOVB    R1,(R5)+                    ; Store CTLFUNC field
                     0071    230
                     0071    231 ;
                     0071    232 ; Map user specified FOP options to be performed on close into the DAP FOP
                     0071    233 ; field. The IFAB bookeepping field contains those FOP options requested on
                     0071    234 ; open/create combined (logically ORed) with those requested on close.
                     0071    235 ;
                     0071    236 ; In addition, the DLT, SCF, and SPL bits (treated as a set) will be mapped iff
                     0071    237 ; partner supports all of the requested options; else ignored for now--they
                     0071    238 ; will be re-examined after Access Complete messages have been exchanged for
                     0071    239 ; potential alternate processing using a separate DAP access function.
                     0071    240 ;
                     0071    241
      51    04 A9    D0       0071    242         MOVL    IFB$L_BKPBITS(R9),R1    ; Get IFAB bits
            52       D4       0075    243         CLRL    R2                      ; Zero resultant FOP bits
      08 67          34       0077    244         BBC     #DAP$V_VAXVMS,(R7),10$  ; Branch if partner is not VAX/VMS
                     007B    245         $MAPBIT <IFB$V_RWC-32>,DAP$V_RWC; Map RWC bit
                     0083    246 ; ##### $MAPBIT <IFB$V_DMO-32>,DAP$V_DMO; Map DMO bit
                     0083    247                                             ; Note: this is not implemented in RMS32
                     0083    248
                     0083    249         ASSUME  IFB$V_SPL+1 EQ IFB$V_SCF
                     0083    250         ASSUME  IFB$V_SCF+1 EQ IFB$V_DLT
                     0083    251
                     0083    252         ASSUME  DAP$V_FOPSPL+1 EQ DAP$V_FOPSCF
                     0083    253         ASSUME  DAP$V_FOPSCF+1 EQ DAP$V_FOPDLT
                     0083    254
56    51    03 09    EF       0083    255 10$:    EXTZV   #<IFB$V_SPL-32>,#3,R1,R6; Extract the three FOP bits
                  23  13       0088    256         BEQL    20$                 ; Branch if none set
            03 1D    EF       008A    257         EXTZV   #DAP$V_FOPSPL,#3,-  ; Extract the three corresponding
      50    28 A7             008D    258                 DAP$Q_SYSCAP(R7),R0 ;  SYSCAP bits
            56 50    CA       0090    259         BICL2   R0,R6               ; Mask out supported requests
                  18  12       0093    260         BNEQ    20$                 ; Branch if any are not supported
                     0095    261         $MAPBIT <IFB$V_DLT-32>,DAP$V_DLT; Map DLT bit
                     009D    262         $MAPBIT <IFB$V_SCF-32>,DAP$V_SCF; Map SCF bit
                     00A5    263         $MAPBIT <IFB$V_SPL-32>,DAP$V_SPL; Map SPL bit
```

```
                51    52   D0  00AD   264 20$:    MOVL    R2,R1                       ; Move data to correct register
                   FF4D'   30  00B0   265          BSBW    NT$CVT_BN4_EXT              ; Store FOP as an extensible field
                           00B3   266
                           00B3   267 ;
                           00B3   268 ; Include file level CRC checksum in message as required.
                           00B3   269 ;
                           00B3   270
                      15   E1  00B3   271          BBC     #DAP$V_DAPCRC,-             ; Branch if partner does not support
                08 28 A7           00B5   272                  DAP$Q_SYSCAP(R7),30$    ;  file level CRC checksum
              04 67    1A   E0  00B8   273          BBS     #NWA$V_FTM_RETRV,(R7),30$;Branch if file transfer retrieval mode
                           00BC   274                                                  ; has not been properly terminated
              85    20 A7   B0  00BC   275          MOVW    DAP$L_CRC_RSLT(R7),(R5)+; Store CRC checksum field (CHECK)
                   FF3D'   30  00C0   276 30$:     BSBW    NT$BUILD_TAIL               ; Finish building message
                   FF3A'   30  00C3   277          BSBW    NT$TRANSMIT                 ; Send Access Complete message to FAL
                  0C 50    E9  00C6   278          BLBC    R0,EXIT1                    ; Branch on failure
                           00C9   279
                           00C9   280 ;+
                           00C9   281 ; Receive Access Complete message from partner (which may be preceded by other
                           00C9   282 ; DAP messages in the pipe if we've prematurely terminated the access).
                           00C9   283 ;-
                           00C9   284
                           00C9   285 RECV_CMP:                                       ;
                   FF34'   30  00C9   286          BSBW    NT$LOOK_FOR_CMP             ; Wait for response from FAL
                  07 50    E9  00CC   287          BLBC    R0,CHKCRC                   ; Branch on failure
                     56    D5  00CF   288          TSTL    R6                          ; Branch if there are no more FOP
                     02    13  00D1   289          BEQL    EXIT1                       ;  options to process
                     31    10  00D3   290          BSBB    REEXAMINE_FOP               ; Try to process them another way
                           05  00D5   291 EXIT1:   RSB                                 ; Exit with RMS cc e in R0
                           00D6   292
                           00D6   293 ;
                           00D6   294 ; Special case DAP level CRC checksum error reported by FAL.
                           00D6   295 ;
                           00D6   296
          0000'8F    50   B1  00D6   297 CHKCRC: CMPW    R0,#<RMS$_CRC&^XFFFF>       ; Check for RMS$_CRC error from FAL
                     02   12  00DB   298          BNEQ    10$                         ;
                     01   10  00DD   299          BSBB    CRC_ERROR                   ; Process DAP level CRC checksum error
                          05  00DF   300 10$:     RSB                                 ; Exit with RMS code in R0
```

```
                     00E0   302          .SBTTL  CHECK_CRC_ERROR
                     00E0   303
                     00E0   304 ;++
                     00E0   305 ; This routine processes an RMS$_CRC error returned by FAL on a close operation.
                     00E0   306 ; First, a DAP level CRC checksum error message is formatted and sent to the
                     00E0   307 ; DECnet Event Logger. Then another Access Complete message is generated to
                     00E0   308 ; direct FAL to close the file.
                     00E0   309 ;--
                     00E0   310
                     00E0   311 CRC_ERROR:                                   ;
       01    BB      00E0   312          PUSHR   #^M<R0>                    ; Save completion code on entry
   FF1B'     30      00E2   313          BSBW    NT$CRC_LOGERR              ; Send message to DECnet Event Logger
                     00E5   314                                             ; No status code is returned
                     00E5   315
                     00E5   316 ;+
                     00E5   317 ; Build and send Access Complete message to close the file.
                     00E5   318 ;-
                     00E5   319
                     00E5   320 CRC_SEND_CMP:                                ;
                     00E5   321          $SETBIT #NMA$V_LAST_MSG,(R7)       ; Declare this last message to block
50     07    D0      00E9   322          MOVL    #DAP$K_CMP_MSG,R0          ; Get message type value
   FF11'     30      00EC   323          BSBW    NT$BUILD_HEAD              ; Construct message header
85     01    90      00EF   324          MOVB    #DAP$K_CLOSE,(R5)+         ; Store CTLFUNC field
   FF0B'     30      00F2   325          BSBW    NT$BUILD_TAIL              ; Finish building message
   FF08'     30      00F5   326          BSBW    NT$TRANSMIT                ; Send Access Complete message to FAL
 08 50       E9      00F8   327          BLBC    R0,CRC_EXIT                ; Branch on failure
                     00FB   328
                     00FB   329 ;+
                     00FB   330 ; Receive Access Complete message from partner.
                     00FB   331 ;-
                     00FB   332
                     00FB   333 CRC_RECV_CMP:                                ;
                     00FB   334          $SETBIT #DAP$K_CMP_MSG,DAP$L_MSG_MASK(R7)
                     0100   335                                             ; Expect response of Access Complete msg
FEFD'     30         0100   336          BSBW    NT$RECEIVE                 ; Get reply from FAL
                     0103   337 CRC_EXIT:                                   ;
       01    BA      0103   338          POPR    #^M<R0>                    ; Restore completion code on entry
             05      0105   339          RSB                                ; Exit with RMS$_CRC code in R0
```

```
                     0106   341          .SBTTL  REEXAMINE_FOP
                     0106   342
                     0106   343 ;++
                     0106   344 ; This routine attempts to process selected FOP options via a separate DAP
                     0106   345 ; access function (which have not been processed because the remote FAL does not
                     0106   346 ; support passing these options in the DAP FOP field in the Access Complete
                     0106   347 ; message).
                     0106   348 ;--
                     0106   349
                     0106   350 REEXAMINE_FOP:                          ; Entry point
                     0106   351
                     0106   352          ASSUME  IFB$V_SPL+1 EQ IFB$V_SCF
                     0106   353          ASSUME  IFB$V_SCF+1 EQ IFB$V_DLT
                     0106   354
     50   69   03   29  EF 0106   355          EXTZV   #IFB$V_SPL,#3,(R9),R0   ; Extract the three FOP bits
                     010B   356          $CASEB  SELECTOR=R0-                ; Options requested:
                     010B   357                  DISPL=<-                    ; Note: SCF takes precedence over SPL
                     010B   358                      NOTHING-                ;   0 ! 0 ! 0
                     010B   359                      ERRSUP-                 ;   0 ! 0 !SPL
                     010B   360                      EXECUTE-                ;   0 !SCF! 0
                     010B   361                      EXECUTE-                ;   0 !SCF!SPL ==> SCF
                     010B   362                      DELETE-                 ;  DLT! 0 ! 0
                     010B   363                      ERRSUP-                 ;  DLT! 0 !SPL
                     010B   364                      SUBMIT-                 ;  DLT!SCF! 0
                     010B   365                      SUBMIT-                 ;  DLT!SCF!SPL ==> SCF!DLT
                     010B   366                  >                           ;
                     011F   367 NOTHING:RMSSUC                            ; Return success
                  05 0122   368          RSB                               ; Exit with RMS code in R0
            FEDA' 31 0123   369 ERRSUP: BRW     NT$RMT_FOP2                ; Declare RMS$_SUPPORT error and
                     0126   370                                            ;  exit with RMS code in R0
               FB 11 0126   371 SUBMIT: BRB     ERRSUP                     ; It's too late to perform DAP submit
                     0128   372                                            ;  function--logically it's similar to
                     0128   373                                            ;  create function
            56 04 90 0128   374 DELETE: MOVB    #DAP$K_ERASE,R6            ; Save DAP function code for erase
               03 11 012B   375          BRB     REEX_SEND_ACC              ; Delete the file
            56 08 90 012D   376 EXECUTE:MOVB    #DAP$K_EXECUTE,R6          ; Save DAP function code for execute
                     0130   377
                     0130   378 ;+
                     0130   379 ; Build and send Access message to partner.
                     0130   380 ;-
                     0130   381
                     0130   382 REEX_SEND_ACC:                            ;
                     0130   383          $SETBIT #NWA$V_LAST_MSG,(R7)      ; Declare this last message to block
            50 03 D0 0134   384          MOVL    #DAP$K_ACC_MSG,R0         ; Get message type value
               FEC6' 30 0137   385          BSBW    NT$BUILD_HEAD          ; Construct message header
            85 56 90 013A   386          MOVB    R6,(R5)+                  ; Store ACCFUNC field
            85 01 90 013D   387          MOVB    #DAP$M_NONFATAL,(R5)+     ; Store ACCOPT field
         5A 38 A9 D0 0140   388          MOVL    IFB$L_FWA_PTR(R9),R10     ; Get address of FWA (no need to save
                     0144   389                                            ;  R10 as it was equal to R9 on entry)
               FEB9' 30 0144   390          BSBW    NT$GET_FILESPEC           ; Store FILESPEC as a counted
                     0147   391                                            ;  ASCII string
            5A 59 D0 0147   392          MOVL    R9,R10                    ; Restore IFAB address to register
               FEB3' 30 014A   393          BSBW    NT$BUILD_TAIL             ; Finish building message
               FEB0' 30 014D   394          BSBW    NT$TRANSMIT               ; Send Access message to FAL
            08 50 E9 0150   395          BLBC    R0,EXIT2                  ; Branch on failure
                     0153   396
                     0153   397 ;+
```

```
                     0153   398 ; Receive Access Complete message from partner.
                     0153   399 ;-
                     0153   400
                     0153   401 REEX_RECV_CMP:                                ;
                     0153   402         $SETBIT #DAP$K_CMP_MSG,DAP$L_MSG_MASK(R7)
                     0158   403                                               ; Expect response of Access Complete msg
FEA5' 30             0158   404         BSBW    NT$RECEIVE                    ; Get reply from FAL
      05             015B   405 EXIT2:  RSB                                   ; Exit with RMS code in R0
                     015C   406
                     015C   407         .END                                  ; End of module
```

```
$$.PSECT_EP                     =  00000000
$$COUNT                         =  00000008
$$RMSTEST                       =  0000001A
$$RMS_PBUGCHK                   =  00000010
$$RMS_TBUGCHK                   =  00000008
$$RMS_UMODE                     =  00000004
BUILD_MASK                         0000000A  R      01
CHKCRC                             000000D6  R      01
CRC_ERROR                          000000E0  R      01
CRC_EXIT                           00000103  R      01
CRC_RECV_CMP                       000000FB  R      01
CRC_SEND_CMP                       000000E5  R      01
DAP$B_ACCFUNC                      00000040
DAP$B_ACCOPT                       00000041
DAP$B_BITCNT                       00000035
DAP$B_BKS                          00000050
DAP$B_BSZ                          00000052
DAP$B_CMPFUNC                      00000040
DAP$B_DATATYPE                     00000044
DAP$B_DCODE_FID                    00000019
DAP$B_DCODE_MAC                    0000001B
DAP$B_DCODE_MSG                    0000001A
DAP$B_DECVER                       00000047
DAP$B_ECONUM                       00000045
DAP$B_FAC                          00000042
DAP$B_FILESYS                      00000043
DAP$B_FLAGS                        00000031
DAP$B_FSZ                          00000051
DAP$B_LEN256                       00000034
DAP$B_LENGTH                       00000033
DAP$B_ORG                          00000045
DAP$B_OSTYPE                       00000042
DAP$B_RAT                          00000047
DAP$B_RFM                          00000046
DAP$B_SHR                          00000043
DAP$B_STREAMID                     00000032
DAP$B_TYPE                         00000030
DAP$B_USRNUM                       00000046
DAP$B_USRVER                       00000048
DAP$B_VERNUM                       00000044
DAP$B_X_FIELD                      00000024
DAP$C_BLN                          000000C0
DAP$K_ACC_MSG                   =  00000003
DAP$K_BLN                          000000C0
DAP$K_CHANGE_B                  =  00000006
DAP$K_CHANGE_E                  =  00000007
DAP$K_CLOSE                     =  00000001
DAP$K_CMP_MSG                   =  00000007
DAP$K_ERASE                     =  00000004
DAP$K_EXECUTE                   =  00000008
DAP$K_FIX                       =  00000001
DAP$K_SEQ                       =  00000000
DAP$L_ALQ1                         0000004C
DAP$L_ATTMENU                      00000040
DAP$L_CMWA                         00000030
DAP$L_CRC_RSLT                     00000020
DAP$L_DCODE_STS                    00000018
DAP$L_DEV                          00000068
DAP$L_EBK                          00000078
DAP$L_FOP1                         00000064
DAP$L_FOP2                         00000044
DAP$L_HBK                          00000074
DAP$L_MRN                          00000058
DAP$L_MSG_MASK                     0000001C
DAP$L_SBN                          0000007C
DAP$L_SSPWA                        00000080
DAP$L_TEMP                         00000090
DAP$M_BITCNT                    =  00000008
DAP$M_CMPFMT                    =  00000008
DAP$M_DMO                       =  00002000
DAP$M_DSP_3NAM                  =  00000200
DAP$M_DSP_ALL                   =  00000004
DAP$M_DSP_ATT                   =  00000001
DAP$M_DSP_KEY                   =  00000002
DAP$M_DSP_SUM                   =  00000008
DAP$M_EMBEDDED                  =  00000010
DAP$M_GET                       =  00000002
DAP$M_GO_NOGO                   =  00000010
DAP$M_IMAGE                     =  00000002
DAP$M_LSA                       =  00000040
DAP$M_MACY11                    =  00000080
DAP$M_MSE                       =  00000010
DAP$M_NONFATAL                  =  00000001
DAP$M_SEGMENT                   =  00000040
DAP$M_TMP1$                     =  000000C0
DAP$M_TMP2$                     =  0000FC00
DAP$M_TMP3$                     =  00020000
DAP$M_TMP4$                     =  01000000
DAP$M_TMP5$                     =  F0000000
DAP$M_ZERO                      =  00000080
DAP$Q_DCODE_FLG                    00000000
DAP$Q_FILESPEC                     00000044
DAP$Q_MSG_BUF1                     00000008
DAP$Q_MSG_BUF2                     00000010
DAP$Q_PASSWORD                     00000050
DAP$Q_RUNSYS                       0000005C
DAP$Q_SYSCAP                       00000028
DAP$Q_SYSPEC                       00000038
DAP$V_DAPCRC                    =  00000015
DAP$V_DLT                       =  00000016
DAP$V_DSP_PRO                   =  00000005
DAP$V_DSP_TIM                   =  00000004
DAP$V_FOPDLT                    =  0000001F
DAP$V_FOPSCF                    =  0000001E
DAP$V_FOPSPL                    =  0000001D
DAP$V_RWC                       =  00000001
DAP$V_SCF                       =  00000015
DAP$V_SPL                       =  00000014
DAP$V_VAX.MS                    =  00000034
DAP$W_BLS                          00000048
DAP$W_BUFSIZ                       00000040
DAP$W_CHECK                        00000042
DAP$W_DEQ1                         00000054
DAP$W_DISPLAY1                     0000004C
```

DAP$W_FFB	00000072				NWA$L_XLTBUFFLG	0000022C			
DAP$W_LRL	00000070				NWA$L_XLTCNT	00000228			
DAP$W_MRS	0000004A				NWA$L_XLTMAXINDX	00000234			
DAP$W_PARTNER	00000006				NWA$L_XLTSIZ	00000230			
DAP$W_VERSION	00000004				NWA$Q_ACS	00000244			
DELETE	00000128	R		01	NWA$Q_BIGBUF	00000170			
ERRSUP	00000123	R		01	NWA$Q_BLD	0000C0F0			
EXECUTE	0000012D	R		01	NWA$Q_FLG	00000000			
EXIT	0000005B	R		01	NWA$Q_INODE	0000025C			
EXIT1	000000D5	R		01	NWA$Q_IOSB	000000D8			
EXIT2	0000015B	R		01	NWA$Q_LNODE	00000160			
IFB$L_BKPBITS	= 00000004				NWA$Q_LOGNAME	0000023C			
IFB$L_FWA_PTR	= 00000038				NWA$Q_NCB	00000264			
IFB$L_NWA_PTR	= 0000003C				NWA$Q_RCV	000000E0			
IFB$V_DLT	= 0000002B				NWA$Q_SAVE_DESC	00000120			
IFB$V_RWC	= 00000027				NWA$Q_XLTBUF1	0000024C			
IFB$V_SCF	= 0000002A				NWA$Q_XLTBUF2	00000254			
IFB$V_SPL	= 00000029				NWA$Q_XMT	000000E8			
IFB$V_WRTACC	= 00000030				NWA$T_ACSBUF	0000026C			
NOTHING	0000011F	R		01	NWA$T_AUXBUF	000005E0			
NT$BUILD_HEAD	********		X	01	NWA$T_DAP	00000000			
NT$BUILD_TAIL	********		X	01	NWA$T_INODEBUF	000004AC			
NT$CLOSE	00000000	RG		01	NWA$T_ITM_ATTR	00000200			
NT$CRC_LOGERR	********		X	01	NWA$T_ITM_END	00000224			
NT$CVT_BN4_EXT	********		X	01	NWA$T_ITM_LST	00000200			
NT$ENCODE_PRO	********		X	01	NWA$T_ITM_MAXINDX	00000218			
NT$ENCODE_TIM_R	********		X	01	NWA$T_ITM_STRING	0000020C			
NT$GET_FILESPEC	********		X	01	NWA$T_NCBBUF	0000052C			
NT$LOOK_FOR_CMP	********		X	01	NWA$T_NODEBUF	00000169			
NT$RECEIVE	********		X	01	NWA$T_RCVBUF	000001A0			
NT$RMT_FOP2	********		X	01	NWA$T_SCAN	00000100			
NT$SCAN_XABCHN	********		X	01	NWA$T_TEMP	00000120			
NT$TRANSMIT	********		X	01	NWA$T_XLTBUF1	000002AC			
NWA$B_ALLXABCNT	0000011C				NWA$T_XLTBUF2	000003AC			
NWA$B_DAP_RAC	000000C9				NWA$T_XMTBUF	000003C0			
NWA$B_FILESYS	000000C5				NWA$V_FTM_RETRV	= 0000001A			
NWA$B_KEYXABCNT	0000011D				NWA$V_LAST_MSG	= 00000000			
NWA$B_NETSTRSIZ	0000016F				NWA$W_BUILD	000000D2			
NWA$B_NODBUFSIZ	00000168				NWA$W_DAPBUFSIZ	000000CA			
NWA$B_ORG	000000C6				NWA$W_DIR_OFF	000000CC			
NWA$B_OSTYPE	000000C4				NWA$W_DISPLAY	000000D0			
NWA$B_RFM	000000C7				NWA$W_FIL_OFF	000000CE			
NWA$B_RMS_RAC	000000C8				NWA$W_JNLXABJOP	0000011E			
NWA$C_BLN	00000800				PIO$A_TRACE	********		X	01
NWA$K_BLN	00000800				RECV_CMP	000000C9	R		01
NWA$L_ALLXABADR	00000100				REEXAMINE_FOP	00000106	R		01
NWA$L_DATXABADR	00000104				REEX_RECV_CMP	00000153	R		01
NWA$L_DEV	000000C0				REEX_SEND_ACC	00000130	R		01
NWA$L_FHCXABADR	00000108				RMS$_CRC	********		X	01
NWA$L_KEYXABADR	0000010C				SEND_CMP	00000064	R		01
NWA$L_MSG_MASK	000000D4				SEND_CMP_BEGIN	00000021	R		01
NWA$L_PROXABADR	00000110				SEND_CMP_CLOSE	00000061	R		01
NWA$L_RDTXABADR	00000114				SEND_CMP_END	0000005C	R		01
NWA$L_SAVE_FLGS	00000128				SEND_PRO	00000047	R		01
NWA$L_SUMXABADR	00000118				SEND_TIM	00000033	R		01
NWA$L_THREAD	000000FC				SUBMIT	00000126	R		01
NWA$L_XLTATTR	00000238				TPT$L_NTCLOSE	********		X	01

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
NF$NETWORK	0000015C (348.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	00000800 (2048.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.09	00:00:00.50
Command processing	123	00:00:00.60	00:00:03.00
Pass 1	273	00:00:09.60	00:00:22.79
Symbol table sort	0	00:00:01.17	00:00:01.84
Pass 2	88	00:00:01.82	00:00:05.00
Symbol table output	28	00:00:00.24	00:00:01.74
Psect synopsis output	2	00:00:00.00	00:00:00.08
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	548	00:00:13.52	00:00:34.99

The working set limit was 1350 pages.
47867 bytes (94 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 802 non-local and 14 local symbols.
407 source lines were read in Pass 1, producing 14 object records in Pass 2.
24 pages of virtual memory were used to define 23 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	15
_255DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	19

1066 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:NTOCLOSE/OBJ=OBJ$:NTOCLOSE MSRC$:NTOCLOSE/UPDATE=(ENH$:NTOCLOSE)+LIB$:RMS/LIB
```

NT0ACCESS
LIS

NT0CLOSE
LIS

NT0BLDXAB
LIS

NT0CONN
LIS

NT0CREATE
LIS

NT0DAPIO
LIS

NT0DAPCRC
LIS

NT0ACCFIL
LIS

NT0BLKIO
LIS